CAPÍTULO 1 – INTRODUCCIÓN A LOS COMPILADORES

1. LENGUAJES DE PROGRAMACIÓN

- Notaciones para describir cálculos.
- Todo software se escribe en un lenguaje de programación.
- Deben traducirse para ser ejecutados.

2. COMPILADORES

- Sistemas de software que traducen un programa fuente a un programa destino.
- Función: Reportar errores durante la traducción.
- Estructura:
  - Front-end: Análisis (léxico, sintáctico, semántico).
  - Back-end: Síntesis (generación de código intermedio, optimización, generación de código destino).

3. INTÉRPRETES

- Ejecutan directamente el programa fuente con las entradas dadas.
- Ventaja: Mejores diagnósticos de errores.
- Desventaja: Menor velocidad que un compilador.

4. PROCESADORES HÍBRIDOS

1. Compilador (Traductor)
Convierte el programa fuente (en Java) en un formato intermedio llamado bytecodes.

2. Código Intermedio (Bytecodes)
Representación intermedia del programa, independiente de la máquina física.

3. Máquina Virtual (Intérprete)
Ejecuta (interpreta) los bytecodes línea por línea.

4. Compilador Just-in-Time (JIT)
Traduce los bytecodes a código máquina nativo justo antes de la ejecución.

5. FASES DE UN COMPILADOR

1.Análisis Léxico (Scanner):
Convierte caracteres en lexemas → tokens.
Ej: posición → <id, 1>

2.Análisis Sintáctico (Parser):
Construye árbol sintáctico a partir de tokens. Define estructura gramatical.

3.Análisis Semántico: Verifica consistencia semántica y tipos.
Realiza coerciones (ej: int → float).

4.Generación de Código Intermedio: Código de tres direcciones (ej: t1 = id3 * 60.0).

5.Optimización de Código: Mejora el código intermedio (velocidad, tamaño, consumo).

6.Generación de Código: Convierte código intermedio en código máquina. Asigna registros y memoria.

6. TABLA DE SÍMBOLOS

- Estructura que almacena información de identificadores (nombre, tipo, alcance, etc.).
- Usada en todas las fases del compilador.

7. HERRAMIENTAS DE CONSTRUCCIÓN

- Generadores de parsers.
- Generadores de scanners.
- Motores de traducción orientados a sintaxis.
- Generadores de generadores de código.
- Motores de análisis de flujo de datos.
- Kits de herramientas integradas.

8. EVOLUCIÓN DE LOS LENGUAJES DE PROGRAMACIÓN

1ª generación: Lenguaje máquina.
2ª generación: Ensamblador.
3ª generación: Alto nivel (Fortran, Cobol, C, Java).
4ª generación: Específicos (SQL, PostScript).
5ª generación: Lógica y restricciones (Prolog).

- Clasificación por paradigma:
- Imperativo (C, Java).
- Declarativo (ML, Haskell, Prolog).
- Orientado a objetos (C++, Ruby).
- Scripting (Python, JavaScript).

9. RELACIÓN ENTRE LENGUAJES Y COMPILADORES

- Los avances en lenguajes exigen nuevos algoritmos de compilación.
- Los compiladores promueven el uso de lenguajes de alto nivel.
- Son esenciales para aprovechar hardware de alto rendimiento.

10. DESAFÍOS EN LA CONSTRUCCIÓN DE COMPILADORES

-Programas extensos y complejos.
-Traducción correcta de infinitos programas posibles.
-Generación de código óptimo es un problema indecidible.
-Equilibrio entre teoría y práctica.